IJCSC 0973-7391 Volume 6 • Number 1 Sep - Mar 2015 pp.67-72 available online at www.csjournalss.com

A Review of Multiprocessor Directed Acyclic Graph (DAG) Scheduling Algorithms

Shivani Sachdeva[1], Poonam Panwar[2]
[1]M.Tech Student, [2]Assistant Professor,
Deptt. of Comp. Sc. & Engg, Ambala College of Engg and Applied Research, Ambala, India.
shivani.sachdeva91@gmail.com, rana.poonam1@gmail.com

Abstract: Scheduling parallel tasks on multiprocessor systems is to allocate a set of tasks to processors such that the optimal usage of processors and accepted computation time for scheduling algorithm are obtained. In multiprocessor systems, an efficient scheduling of a parallel program onto the processors that minimizes the entire execution time is vital for achieving a high performance. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking. A multiprocessor system can be homogeneous (i.e. having all processors of same capability) or heterogeneous (i.e. having processors of different capabilities) nature. In literature several algorithms are proposed to find optimal solution of multiprocessor task scheduling algorithms. So, in present study we have reviewed various multiprocessor scheduling approaches to find their drawbacks so that an efficient heuristic based algorithm can be developed for obtaining optimal solution for these problems.
Index Terms: Directed Acyclic Graph (DAG), Scheduling, Genetic Algorithm (GA), Selection, Crossover, Mutation.

I. INTRODUCTION

The scheduling in multiprocessors systems is assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. This is usually done to load balance and share system resources effectively or achieve a target quality of service. Multiprocessor scheduling algorithm is typically to schedule all the subtasks on a given number of available processors in order to minimize makespan without violating precedence constraints [1]. A multiprocessor task scheduling problem can be represented as directed acyclic task graph (DAG), for execution on multiprocessors with communication costs [2].

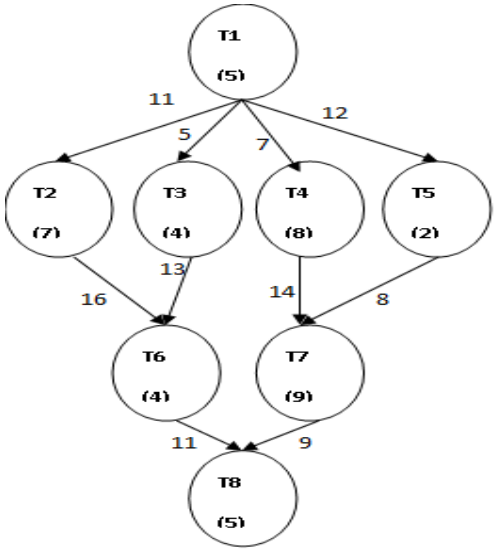

Figure 1. . An Example of DAG

Directed Acyclic Graph (DAG)
A directed acyclic graph (DAG) is a directed graph with no cycles. It is formed by a collection of vertices and directed edges. DAGs may be used to model many different kinds of information. A collection of tasks that are ordered into a sequence, subject to constraints that some tasks must be performed earlier than others, may be

represented as a DAG with a vertex for each task and an edge for each constraint. An application can be represented by a DAG as in Figure 1, with the vertices representing subtasks and edges between vertices representing execution precedence between subtasks. A weight is associated with each vertex and edge. In Figure 1, the vertex weight represents the amount of computation to be performed by the subtask T_i, while the edge weight represents the amount of communication between subtasks $T_i$and subtask T_i. Our goal is to find a schedule which is a mapping of tasks to processors that minimizes the makespan. A parallel and distributed computing system could be a homogeneous or heterogeneous system:

Homogeneous Multiprocessors

Scheduling in homogeneous multiprocessor systems is an important computing problem, because task scheduling problem is an NP-Hard problem. Homogeneous computing refers to the system that uses same kind of processors to minimize possible time required to schedule all jobs on similar type of processors. Minimizing the execution time of the parallel program is the aim of scheduling. Scheduling is done so that any task can process with minimum waiting time for processing tasks that dependent to them, according to dependencies between tasks in the task graph. The critical aim of a scheduler is thus to assign partitioned tasks to available processors in a manner such that:

1. The requirements (or constraints) of precedence between these tasks are met.
2. The resulting overall length of time required to execute the entire program, the schedule length or make span, is minimized [3, 4].

Heterogeneous Multiprocessors

Heterogeneous computing refer to the system that uses different kind of processors to minimize possible time required to schedule all jobs on different types of processors. Heterogeneous systems have different processing capabilities in the target system. In general, the processors are connected by a network, which is either fully connected or partially connected. One of the key disadvantages of heterogeneous multiprocessor design is the need to use a different software development tool set (operating system, compiler, assembler, debugger, instruction-set simulator, etc.) for each of the different processors used in the system design [5, 6, 7].

II. VARIOUS TECHNIQUES TO SOLVE MULTIPROCESSORS SCHEDULING PROBLEMS

There are deterministic approaches and non-deterministic approaches for solving the task-scheduling problem:

Deterministic Approaches

An algorithm is referred to as a deterministic algorithm if given a specific input it will always produce the same output and the underlying machine always passes through the same sequence of states. Most of the deterministic algorithms are based on a greedy strategy, which merely attempts to minimize the finishing time of the tasks in such a way that tasks are allocated to the parallel processors without backtracking. Scheduling is usually handled by heuristic methods algorithms are list scheduling algorithm, clustering algorithm, and task duplication algorithm. However, these deterministic algorithms can only solve certain cases efficiently and cannot preserve the consistent performance on a broad range of problems due to the greedy property.

-List scheduling algorithm

The search in list scheduling algorithms is divided into two phases: In the phase first, each subtask is assigned a priority and then added to a list of waiting subtasks in order of decreasing priority according to certain criteria. In the phase second, as processors become available, the subtask with the highest priority is selected and assigned to the processor e.g. MH [8], Time placement problem [9], HEFT [10,11], CPOP [10,11].

-Clustering algorithm

These algorithms assume that there is enough number of processors available to the subtask execution. In the clustering algorithms use as many processors as possible in order to reduce the makespan of the schedule e.g. A k-means Clustering algorithm [12].

-Task duplication algorithm

It attempts to reduce communication delays by executing a key subtask on more than one processor e.g. Optimal Scheduling [13].

Non-Deterministic Approaches

A non-deterministic algorithm differs from a deterministic algorithm in its capability to produce various outcomes depending on the choices that it makes during execution. Currently, non-deterministic algorithms have been used to solve a wide range of combinatorial problems, which take more time to explore the solution space for a high-quality solution. To solve the scheduling problem, several research studies have been performed based on non-deterministic algorithms, such as genetic algorithms, ant colony optimization algorithms (ACO), and particle swarm optimization algorithms (PSO), artificial bee colony algorithm (ABC), simulated annealing (SA), cuckoo search algorithm (CS), tabu search (TS), evolution algorithm which apply randomized search techniques in the search process.

-Particle Swarm Optimization (PSO)

PSO is a population based search algorithm based on the simulation of birds with in a flock. The initial intent of the particle Swarm concept was to graphically simulate the graceful and unpredictable choreography of a bird flock, with the objective of discovering patterns that govern the ability of birds to fly synchronously and to suddenly change direction with a regrouping in an optimal formation e.g. In Electromagnetics [14].

-Ant Colony Optimization (ACO)

ACO is inspired by behaviour of real ants moving about in search of food. Ants while moving in search of food deposit a pheromone along the path they move with the objective of selecting a path with greater amount of pheromone by new ant e.g. Routing and load balancing [15].

-Tabu Search

Tabu Search is an iterative neighbourhood search algorithm, where the neighbourhood changes dynamically. Tabu Search enhances local search by actively avoiding points in the search space already visited e.g. p-Median Problem [16].

-Simulated Annealing

Simulated Annealing is an optimization process based on the physical process of formation of crystals in solids while cooling. Simulated Annealing is an algorithm implementation of the cooling process to find the optimum of an objective function e.g. Transmission System Expansion Planning [17].

-Random Search

Random search is an extremely basic method. This method only explores the search space by randomly selecting solutions and evaluates their fitness. This is quite an unintelligent strategy, and is rarely used by itself. It doesn't take much effort to implement it and an important number of evaluations can be done fairly quickly. For new unresolved problems, it can be useful to compare the results of a more advanced algorithm to those obtained just with a random search for the same number of evaluations e.g. Global Optimization [18].

-Genetic Algorithm (GA)

In our work we are planning to use GA, so GA are discussed in some detail in task scheduling. The term genetic algorithm stands for a broad class of algorithms, which exploits an analogy to natural evolution. There is a set of individuals called population, which is initialized at random and maintained according to the "survival of the fittest" principle. Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization problems. GAs are based on the principle of evolution and genetics. GAs is used to resolve complicated optimization problems, such as, job-shop scheduling, timetabling, games playing, etc. GA handles a population of possible solutions. Each solution in GA are represented through a chromosome. In GA coding of all the possible solutions into a chromosome is the first phase, but certainly not the most straightforward one of a Genetic Algorithm (GA). A set of reproduction operators has to be determined, too. On the chromosomes, reproduction operators are applied directly and are used to perform mutations and recombination's over solutions of the problem. For genetic algorithm, a randomly generated initial population of search nodes is required. The flowchart showing the process of GA is shown as in Figure 2 [19].

The basic genetic algorithm is as follows [19]:

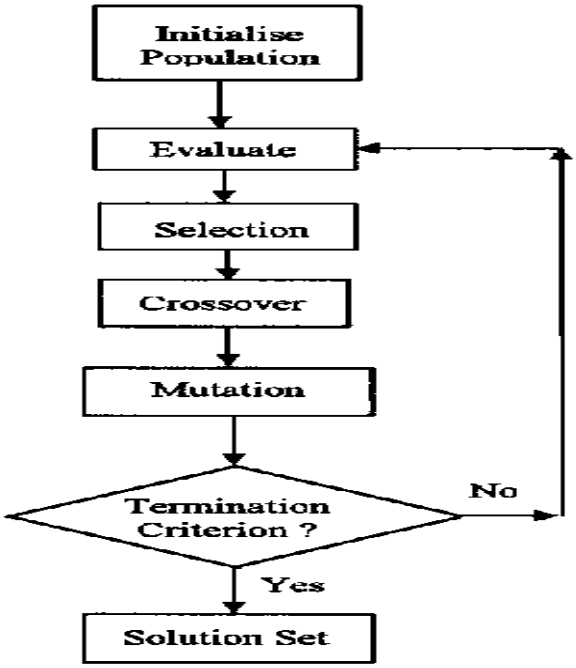

Figure 2. Working of Genetic Algorithm

Initial Population
Start with a randomly generated population of n, l-bits chromosomes which are initially selected candidate solutions i.e. called gene pool.

Fitness Function
Calculate fitness function f(x) of each chromosome x in the gene pool. Multiprocessor scheduling problem will also consider factors such as throughput, finishing time and processor utilization. Genetic algorithm is based on finishing time of a schedule. The finishing time of a schedule, S is defined as follows:

$$FT(S) = \max P_j\ (ftp(P_j))$$

Where, $ftp(P_j)$ is the finishing time for the last task in processor P_j.

Genetic Operators
Function of genetic operators [19] is to create new search nodes based on the current population of search nodes. Various genetic operators are described below.

Selection
It is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection i.e., how to choose individuals in the population that will create offspring for the next generation and how many offspring each will create.

Crossover
It is a recombination operator that proceeds in three steps:

1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

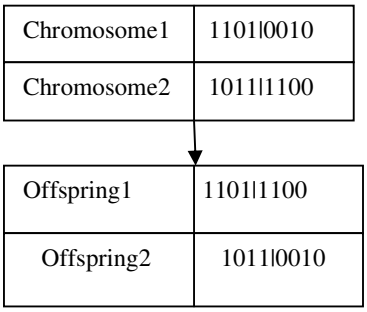

Figure 3 Crossover

Mutation

It alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation of a bit involves flipping a bit, changing 0 to 1 and vice-versa.

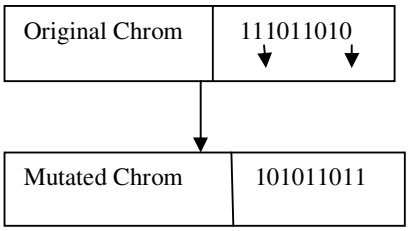

Figure 4 Mutation

Genetic Algorithm in Brief

The genetic algorithm can be summarized in brief as

1. [Initialize]
2. [Fitness] Compute fitness values for each string in the initial population.
3. [New Population] Create a new population by repeating steps 4 to 7 until the new population is complete.
4. Perform Selection
5. Perform Crossover
6. Perform Mutation
7. [Accepting] Place new offspring in the new population and discard old ones having worst fitness value.
8. [Terminate] If the end condition is satisfied, stop.

III. CONCLUSION

In this paper we described various multiprocessor scheduling algorithms proposed in literature which are used to schedule various tasks represented by DAG on multiprocessor systems. We have also discussed briefly various heuristic approaches of scheduling in this paper and in future we are planning to generate an efficient algorithm using GA to schedule task on multiprocessor system for obtaining optimal makespan in minimum time.

IV. REFERENCES

[1] Xu, Yuming, Kenli Li, Jingtong Hu, and Keqin Li. "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues." Information Sciences 270 (2014): 255-287.

[2] Panwar, Poonam, A. K. Lal, and Jugminder Singh. "A Genetic Algorithm Based Technique for Efficient Scheduling of Tasks on Multiprocessor System." In Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011, pp. 911-919. Springer India, 2012.

[3] Dhingra, Sunita, Satinder Bal Gupta, and Ranjit Biswas. "Comparative analysis of heuristics for multiprocessor task scheduling problem with homogeneous processors." Advances in Applied Science Research 5, no. 3 (2014): 280-285.

[4] Kaur, Kamaljit, Amit Chhabra, and Gurvinder Singh. "Modified genetic algorithm for task scheduling in homogeneous parallel system using heuristics."International Journal of Soft Computing 5, no. 2 (2010): 42-51.

[5] Topcuoglu, Haluk, Salim Hariri, and Min-you Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing." Parallel and Distributed Systems, IEEE Transactions on 13, no. 3 (2002): 260-274.

[6] Arabnejad, Hamid, and Jorge G. Barbosa. "List scheduling algorithm for heterogeneous systems by an optimistic cost table." Parallel and Distributed Systems, IEEE Transactions on 25, no. 3 (2014): 682-694.

[7] Goh, Chi Keong, Eu Jin Teoh, and K. C. Tan. "A hybrid evolutionary approach for heterogeneous multiprocessor scheduling." Soft Computing 13, no. 8-9 (2009): 833-846.

[8] H. El-Rewini and T.G. Lewis, "Scheduling Parallel Program Tasks onto Arbitrary Target Machines," J. Parallel and Distributed Computing,vol. 9, no. 2, pp. 138-153, 1990.

[9] Mtibaa, Abdellatif, Bouraoui Ouni, and Mohamed Abid. "An efficient list scheduling algorithm for time placement problem." Computers & Electrical Engineering 33, no. 4 (2007): 285-298.

[10] H. Topcuoglu, S. Hariri, and M. Wu, "Task Scheduling Algorithms for Heterogeneous Processors,"Proc. Eighth Heterogeneous Computing Workshop (HCW),pp. 3-14, 1999.

[11] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274, Mar. 2002.

[12] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Applied statistics (1979): 100-108.

[13] Park, Chan-Ik, and Tae-Young Choe. "An optimal scheduling algorithm based on task duplication." In Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference on, pp. 9-14. IEEE, 2001.

[14] Robinson, Jacob, and Yahya Rahmat-Samii. "Particle swarm optimization in electromagnetics." Antennas and Propagation, IEEE Transactions on 52, no. 2 (2004): 397-407.

[15] Sim, Kwang Mong, and Weng Hong Sun. "Ant colony optimization for routing and load-balancing: survey and new directions." Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 33, no. 5 (2003): 560-572.

[16] Rolland, Erik, David A. Schilling, and John R. Current. "An efficient tabu search procedure for the p-median problem." European Journal of Operational Research 96, no. 2 (1997): 329-342.

[17] Romero, R., R. A. Gallego, and A. Monticelli. "Transmission system expansion planning by simulated annealing." Power Systems, IEEE Transactions on 11, no. 1 (1996): 364-369.

[18] Price, W. L. "Global optimization by controlled random search." Journal of Optimization Theory and Applications 40, no. 3 (1983): 333-348.

[19] Sivanandam, S. N., and S. N. Deepa. Introduction to genetic algorithms. Springer Science & Business Media, 2007.